

is stored as a symmetric binary matrix instead of as a graph, is to square the matrix, binarize the result with a threshold of 2, and AND it with the original matrix. The idempotency of matrices has been discussed, e.g. by Rosenblatt.⁷

One example so far has been worked out. The connected graph chosen had 1000 points and 1791 lines. A single operation M (removing lines which do not belong to even one K3), produced 14 fragments of 413, 298, 134, 81, 18, 12, 9, 7, 6, 5, 5, 3, 3 lines respectively. The four largest fragments were subjected to D, followed by D².

Operator D yielded 15 subgraphs of 3 to 18 lines and 7 larger subgraphs, the largest being of 167 lines. Application of the operator D² to the larger subgraphs yielded 14 subgraphs of 6 to 18 lines (6 being the number of lines in a single K4), and 14 of 20 to 75 lines.

Progress so far appeared reasonably satisfactory: we had 63 small but strongly connected graphs, from which 63 word groups could be easily derived. We still wanted somehow to partition the 14 larger graphs, however.

First the operation T² was tried. It revealed that the original graph included one K6 and twenty K5's (six of which were included in the K6); but it added little information about the small graphs, since a single K5 contains 10 lines.

The graphs remaining, of 20 to 75 lines, have been further partitioned by a method which approximates to the finding of cutpoints, but the work is continuing. Programs are also in use for attaching to the small graphs points (i.e., vocabulary words) which have become totally detached in the fragmentation process.

Semantic content of mechanically generated groups of words

The hope was that these computer programs would yield word groups which would make sense to the lexicographer and, in this particular case, to the radio scientist. They do seem to be highly associated semantically, in spite of the complete absence of any semantic considerations in the processing. As an example, all the groups of 4 to 6 words which have come out of the operations so far are printed below. The words have been left in the stem form originally used.

MG:	extincti incidenz polariz reflect obliqu ellip	function polynomi transfer pole real	crystal exciton vibrat shear mode	detonat explosi nuclear island	reactor saturat core
DG:	ferromag magnetic magnetis magnet spin	germaniu reverse silic rectif	irregul aspect spread align echo	outburst burst erupt flare	plastic stress print adhe bond view
					radiant meteor shower trail

D ² G:	collisic hydrodyn magneto ionic ioniz	electron nitrogen neutral diffus gases ion	discharg pressure gaseous plasma probe	geomagne pulsat storm bay
-------------------	---	---	--	------------------------------------

A phonological test of semantic content is also in hand. An evaluation of the sets of word groups, by systematically using them for information indexing and retrieval, has been embarked on.

Other applications of graph algorithms

At this time, many groups of people, working on complex associations of one form or another, must be aware that their problem could in some way be expressed as the partitioning of large graphs under suitable restraints. It is time that a journal or a society were formed in which they could exchange their views.

ACKNOWLEDGMENT

I would like to thank Dr. P.K.T. Vaswani who initiated the plan for machine generation of word groups for many valuable discussions, and Miss C. A. Walsh for help with preparation of data and programming.

The work described above has been carried out as part of the research programme of the National Physical Laboratory, and this paper is published by permission of the Director of the Laboratory.

REFERENCES

- 1 A R MEETHAM
Preliminary studies for machine-generated index vocabularies
Language and Speech 6 22 1963
- 2 A R MEETHAM
Probabilistic pairs and groups of words in a text
Language and Speech 7 98 1964
- 3 P K T VASWANI
Mechanized storage and retrieval of information
Rev Int Doc 32 19 1965
- 4 M KOCHEN
Information science
The Scarecrow Press Inc New York 1965
- 5 A F PARKER-RHODE R M NEEDHAM
The theory of clumps
CLRU Cambridge England and CFSTI PB 166804
- 6 A R MEETHAM
Algorithm to assist in finding the complete subgraph of a given graph
Nature in press 1966
- 7 D ROSENBLATT
On the graphs and asymptotic forms of finite Boolean relation matrices and stochastic matrices
Naval research logistics quarterly 4 151 1957

Display oriented computer usage system

by HAROLD S. CORBIN and WERNER L. FRANK
Informatics Inc.
Bethesda, Maryland

INTRODUCTION

The history of data processing has shown the development of general purpose packages in support of data processing hardware. Thus, we have seen the evolution of assemblers, compilers, service routines, and I/O routines. These software packages have led us to operating systems, job monitors, and data manipulation packages in support of batch processing operations.

The advent of user-oriented terminal devices and conversation modes of on-line operation, be it time sharing or direct access systems, uncovers new software requirements. Thus new tools are required by application implementors to:

- 1) Describe man/machine conversational procedures through appropriate display console oriented languages,
- 2) Compile these languages into computer operator code,
- 3) Execute this code within a responsive on-line multiple user environment,

and by on-line users to:

- 1) Communicate with the data processing system by problem oriented operations,
- 2) Be able to generate, on-line, additional operations built upon existing operations,
- 3) Manipulate displays.

There have been several efforts, which attempted to develop or are currently developing, man-machine communication techniques. Recognizing the impressive list of existing time-sharing systems, the references identified here are limited to systems using display consoles in a general purpose, conversational or interactive mode with the human operator. In 1960 the system design for DODDAC¹ advanced some basic approaches. An On-Line Computer Center² by Culler-Fried and the related TRW Two-Station On-Line Scientific Computer³ used the man-machine concept for scientific problem solving. Work for the National Military Command System Support Center⁴, SDC's General Purpose Display System⁵ and MITRE's AESOP⁶ are contributing to man-

machine techniques in information retrieval. Sketchpad⁷, General Motors DAC-1 System⁸ and IBM's Alpine System⁹ made contributions in graphical design.

The development of software techniques for on-line information processing and man/machine communication is then the focal point for the Display Oriented Computer Usage System (DOCUS)*. Techniques are being developed to facilitate the implementation of on-line procedures and the operation of such procedures through a man/computer conversation mode. The users' areas of interest may be related to information retrieval, scientific research, management, or command and control. It is desirable to make these on-line users less dependent upon the professional programmer by simplifying the implementation and reducing the time for programming new applications.

DOCUS has been implemented for a CDC 1604B computer to which is attached a Bunker Ramo 85 display console. The techniques which were developed are, however, not restricted to this hardware which has served primarily as a test bed.

Fundamental operations

Current available computers and visual display devices are quite adequate for most of the communication needs of users. However, the user lacks the languages and the operating system to describe procedures and to tie various languages together, to interface with the equipment, and to provide a reference for his operations.

In the DOCUS concept this man/machine communication is accomplished by a series of related steps that user performs an action (pressing a function key on the console, using a light pen, or entering data via a keyboard); the computer responds to the user's actions and presents to him a display or turns on or off some indicator lights; the user examines the computer's message and performs another action. The process continues until the user is satisfied or the computer is unable to respond to the user in a satisfactory manner

*This work is sponsored by the Rome Air Development Center under Contract AF30(602)3500.

and an alarm condition occurs. This operation is diagrammed in Figure 1.**

When solving different problems, there are often programming elements common to these problems. One example is the use of an I/O subroutine. These common elements may be basic logical operations such as *seek a display*, *add an item to a list*, *order a list*, or *read a record*. Operations of this type will be referred to as elementary functions.

In the design phase of an approach, individual elementary functions usually are identified. If enough common functions can be identified and retained in a library, then the solution to a new problem can be simplified by using elementary functions developed during the course of solving previous problems.

In DOCUS these elementary functions can be combined and executed on-line and therefore improve the time required to solve a problem. A second benefit also may accrue to the user of DOCUS. Because the user

**Most of the illustrations were prepared by the use of DOCUS.

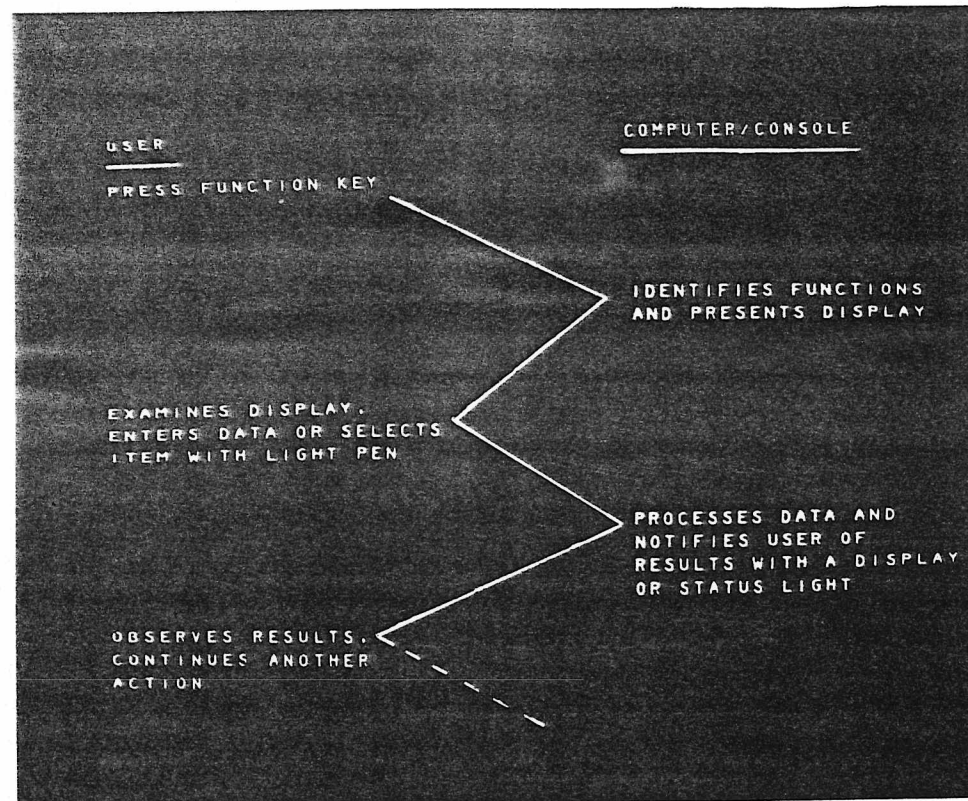


Figure 1 — Man/machine interaction

interacts very closely with the solution as it is developed, he may detect unexpected events and make ad hoc decisions which redirect the approach to the solution.

Within DOCUS, a non-programming user having a sufficiently large number of flexible, elementary functions available in a library, will be less dependent upon the programmer in solving a problem. He will now be "programming" (for want of a better word) the system but he will be doing it in his own application oriented language and not in a machine or compiler language. Toward these ends the basic objectives of DOCUS may be summarized as follows:

- 1) Design new techniques of man/machine communications.
- 2) Improve existing man/machine techniques.
- 3) Develop display console utilization methods.
- 4) Implement and test selected man/machine techniques.

The techniques developed in DOCUS are indepen-

dent of time sharing, multi-programming or multi-processing. It is only when the economics of a system are considered or console-to-console interactions are considered that these modes of operation become important. Since there is much work in progress in these areas, the implementors of DOCUS have borrowed freely from the variety of time sharing and multi-programming methods and concentrated on man/machine aspects.

User aspects

DOCUS recognizes two classes of users: the analyst, and the programmer. The analyst is the ultimate user for whom the hardware/software system exists; he is the problem solver and he is the person who interacts with data. He is least likely to know or care much about details of data processing hardware or software since he looks upon the computer as a tool. Furthermore, he prefers an application oriented language rather than any stylized computer language.

The programmer is a data processing technician who understands the analyst's applications sufficiently so that he can adequately design and implement elementary functions for use by the analyst. As such he will also understand computer programming, the processor hardware and the techniques of man/machine communication.

DOCUS provides two types of software tools to enable the analyst and the programmer to operate more efficiently. First, the programmer has at his disposal a variety of man/machine communication techniques, programming languages and a Display Oriented Language (DOL). The Display Oriented Language is used to express application procedures and implement them for the analyst within the DOCUS operating system. These application procedures may be complete programs for carrying out such functions as querying, editing, or mathematical analysis, or they may consist of data processing functions such as sorting, report generation, and data conversion. However, in most cases these procedures are elementary functions which are display oriented. The Display Oriented Language consists of instructions which govern the step-by-step process of man/machine communication together with necessary operations for creating and manipulating operational elements, which in this case are displays and functions. In addition, the programmer has available debugging and utility programs for on-line use.

Second, the analyst, having at his disposal basic building blocks of elementary functions provided by the programmer, may manipulate these functions on-line by combining, modifying, and organizing them to generate more complicated functions for his use. In this process, he employs terminology and operations familiar to him

Hence, the analyst can expeditiously generate his own personalized processing capability.

Definitions

The following definitions are given for a number of important features and concepts used in discussing display consoles. These are:

Function Key	A switch on the console which causes an interrupt of the data processor and commences the execution of a program or identifies a condition.
Key Program	A group of machine instructions which are executed when a function key is pressed. Usually, a key program is an elementary function, e.g., add item to list or sine X.
Key Group	Several associated or related function keys.
Page	The information, text or graphics, displayed on the CRT at one time.
Display	One or more pages.
Status Light	A light on the console which can be turned on or off under program control.
Light Pen	A pointing device which when pointed at an emitting light source on the CRT provides positioning information to the data processor.
Cursor	A pointing device which is manually controlled to provide positioning information to the data processor.
List Display	A page or display as shown in Figure 2. Each item is preceded by an asterisk. When an item is selected with the light pen the asterisk changes to an X.
Format Display	A page or display as shown in Figure 3. The format space into which data is entered by the use of the Alphanumeric keyboard is specified by underscores. The entry point for the first character is specified by ↓.

Docus components

DOCUS is an open-ended system permitting the addition of new capabilities, represented by application oriented key groups and more programming languages. There are, however, certain basic system components which must always be present. These components form the basic structure of the system and consist of the operating system, the library, and the executive languages — either the General Operating Language (GOL)

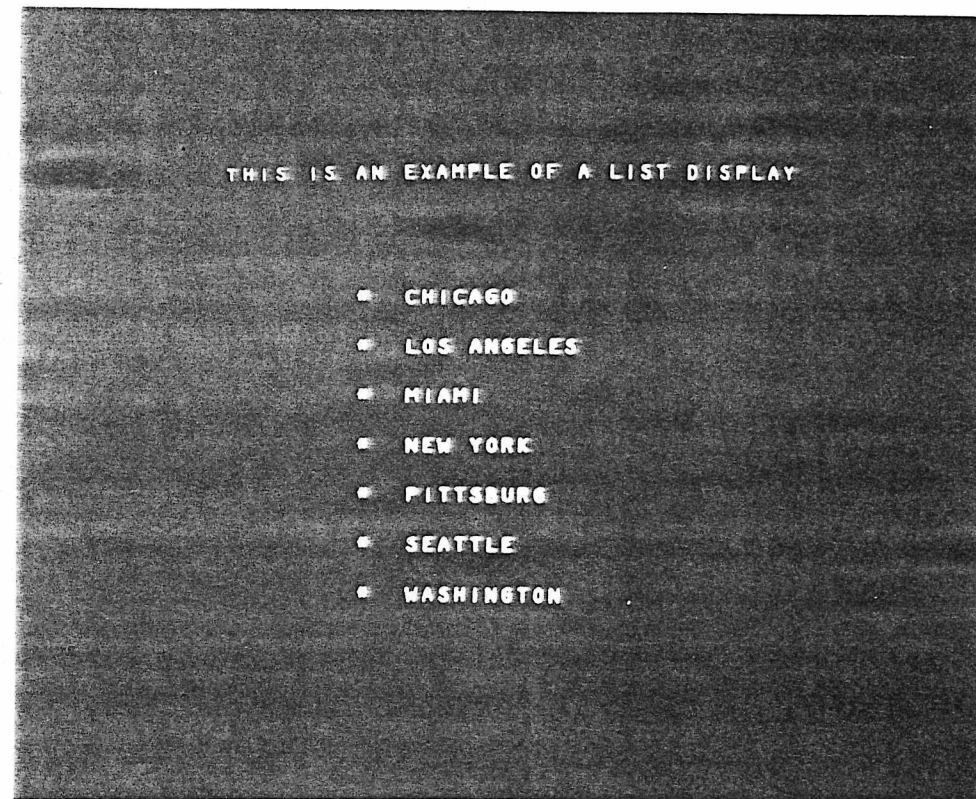


Figure 2 — List display

or the Procedure Implementation Language (PIL). There are, in addition to the basic components, several other components which enhance and demonstrate the DOCUS concept. These include three programming languages, FORTRAN, CODAP, DOL (Display Oriented Language), a Debugging Language, and four representative application key groups for File Management, Text Manipulation, Computation, and Graphics.

Operating system

The operating system serves both types of DOCUS users. For the analyst it provides continuity between his actions and the basic routines concerned with the retrieval of displays and the execution of functions. For the programmer the operating system is the framework under which the implementation cycle for new procedures can be reduced. Incorporated in the system are all of the bookkeeping, transfers, display manipulation, and input/output control normally of concern to the programmer.

The operating system provides the overall control to allow multiple console operation and the execution of a background program during time not required for console operation. It also handles the interrupts from consoles, general display manipulation operations and library processing.

Docus library

The DOCUS library is the principal storage area for function key programs and displays. As such, it is an indirect working storage medium for both the analyst and the programmer (see Figure 4). All function key programs which the analyst executes are initially stored in the library as are the displays required by the analyst. The programmer has the responsibility of building initial library functions for the analyst. The analyst can, however, create certain elements in the library. For example, general purpose elementary function keys which have been combined by the analyst and assigned to a new key could be entries in the analyst's library. Displays which

he has created for his own use may also be stored in the library.

The library is organized on a two-level basis. One part of the library is the general Systems Library (SYSLIB). The SYSLIB contains function key programs and displays which are available for all system users. The other part of the library, Users Own Library (USELIB) is the property of the individual user. A user may access other users' libraries but he may not make additions, changes, or deletions to the other users' USELIB.

System languages

In DOCUS two types of languages are used — executive languages and application languages — to handle common functions related to the general operation of the system and common recurring functions. Executive languages are oriented to the particular type of user. The analyst has one executive language and the programmer has a different executive language. This is

because the two users need to control the system in different ways, e.g., the programmer would need to assemble or compile a program whereas the analyst would not normally need such a function. This, of course, does not mean that an analyst capable of writing a program would be prohibited from doing such a task, but merely that in so doing he has changed his role as a user and would use a different executive language to control the system. The executive language for the analyst is the General Operating Language (GOL), the Procedure Implementation Language (PIL) is for the programmer.

General operating language

The General Operating Language (GOL) provides the basic framework in which the user may solve his problem by the application of functions from either his personal library or the system library. A summary of the keys organized by functional areas is given in Table I.

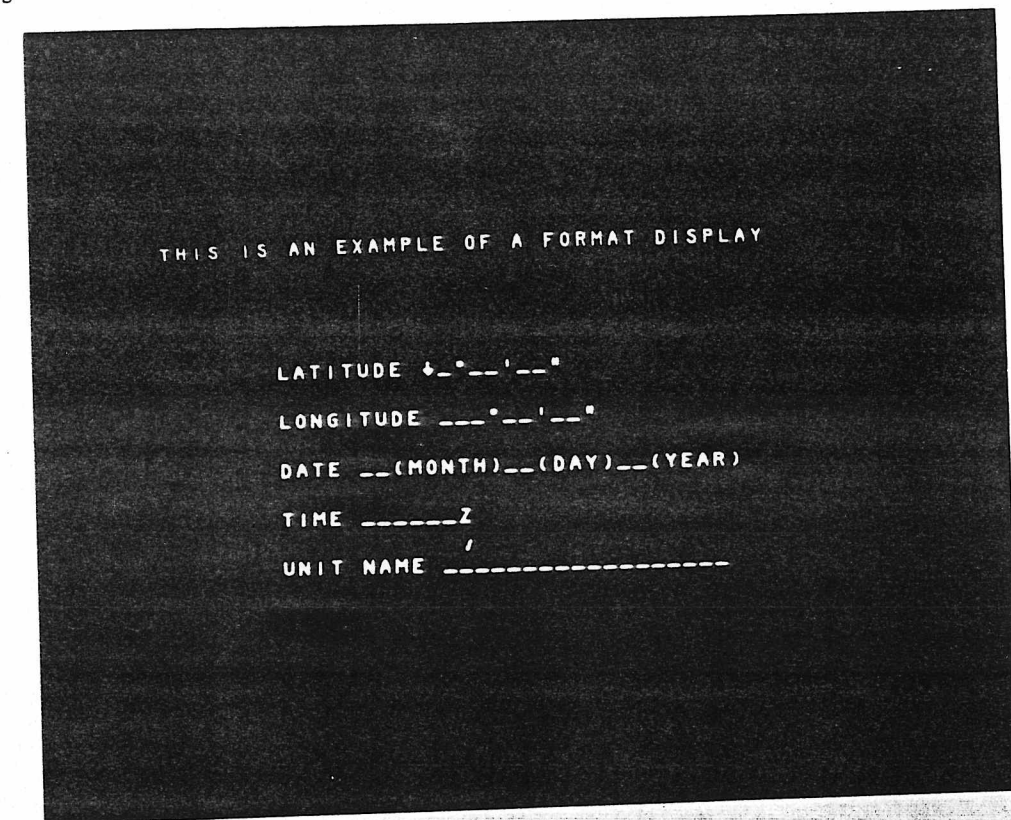


Figure 3 — Format display

<i>General communication</i>	<i>General control</i>
SEND MESSAGE	START/STOP
SHOW MESSAGE	HALT/PROCEED
	SELECT
<i>DOCUS library control</i>	REJECT
LIST DISPLAYS	ENTER
LIST KEY PROGRAMS	
END DISPLAY	<i>Display manipulation</i>
END PAGE	SEEK AND SHOW
CHANGE PAGE	SAVE PAGE
DELETE PAGE	RESTORE PAGE
INSERT PAGE	NEXT PAGE
ADD DISPLAY TO LIBRARY	PREVIOUS PAGE
DELETE DISPLAY FROM LIBRARY	SCAN FORWARD
COMBINE	SCAN BACKWARDS
LIST COMBINE	<i>Output control</i>
	LIST OUTPUT

Table I — General operating language

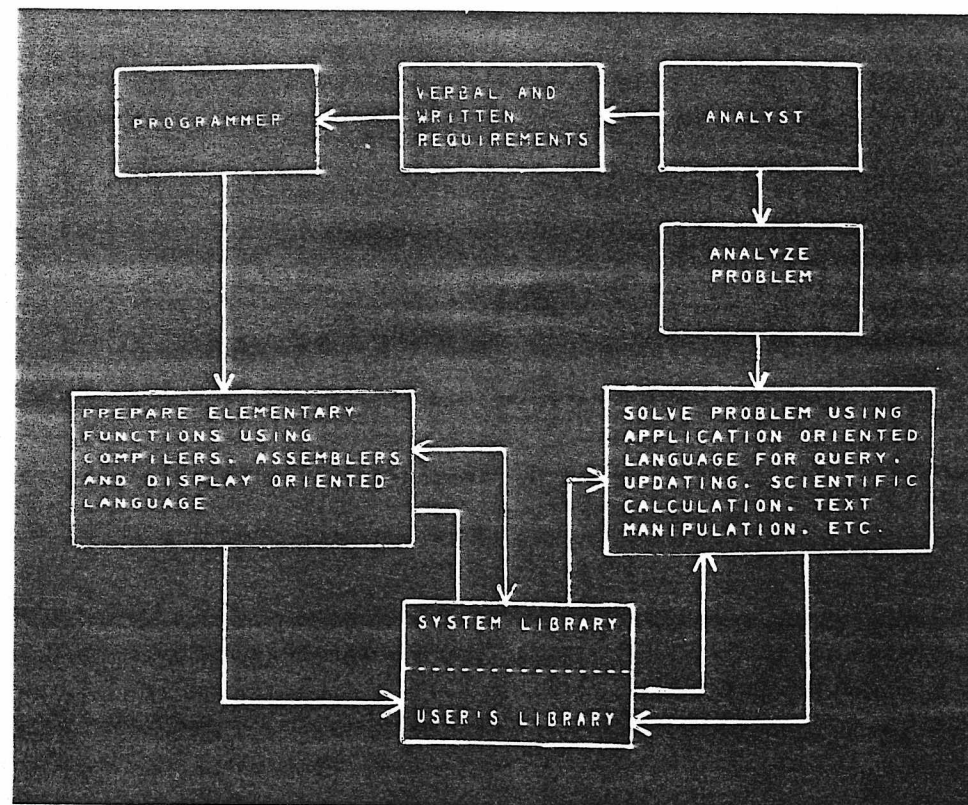


Figure 4 — User and library relationship

The functional characteristics of four selected GOL keys are given below.

SELECT — This key operates in conjunction with the light pen. It allows the user to indicate items of interest within a list display. If the user desires to select entries in a list display, he indicates his intention by pressing SELECT. After this has been accomplished, any item in the current list display may be selected with a light pen. As a result of selecting one or more items, a table of selected items is built by the system.

SHOW MESSAGE — The SHOW MESSAGE key gives the user a method of responding to a "message waiting" status light. This status light is turned on whenever a message appears in the message buffer. The message can be placed there by a key program, the system, or another console. When the console user wants to see the message, he does so by pressing the SHOW MESSAGE key. If he does not want to destroy his current display, he may save the current page with the SAVE PAGE key. Then he presses the SHOW

MESSAGE key and after examining the message he returns to his normal sequence of processing with the RESTORE PAGE key.

LIST DISPLAYS — This key provides the user an up-to-date listing of all current displays residing in the DOCUS library. This display is obtained from the system library are shown and next the names of displays in the various user libraries are shown. The list provides the user with the correct name and number of pages contained within a display. The page control keys: SCAN FORWARD, SCAN BACKWARD, NEXT PAGE and PREVIOUS PAGE may be used to manipulate the list of displays since it is a multi-page display.

DELETE DISPLAY FROM LIBRARY — This key permits the user to remove any previously identified display from the library. Pressing DELETE DISPLAY FROM LIBRARY presents a format display. The user would enter the name of the display with the alphanumeric keyboard and press the ENTER key to execute the function.

Associated with the General Operating Language key group are several status lights which indicate errors or system conditions. Figure 5 shows the status lights used

CONSOLE ON-LINE	LIGHT PEN IN USE			
KEY ERROR	KEY GROUP ERROR	TAPE ERROR	DISPLAY ERROR	KEY PROG. ERROR
LIGHT PEN ERROR	DISC ERROR	SYSTEM LOADER ERROR		
MESSAGE WAITING				PAGE STORE EMPTY

Figure 5 — GOL status lights

with GOL. These lights are programmable and are controlled by the system. They can be reset by operator actions at the console.

• Procedure implementation language

The Procedure Implementation Language (PIL) provides the basic framework in which the programmer

may solve his problem — creating and modifying functions and displays for the analyst. Since certain functions are basic to the operation of DOCUS regardless of the user, they are common to both GOL and PIL. The functions which are unique to PIL are shown in Table II.

BEGIN PROGRAM	KEEP SOURCE PROGRAM
INSERT CODE	COMPILE/ASSEMBLE
DELETE CODE	ADD PROGRAM TO LIBRARY
CHANGE CODE	DELETE PROGRAM FROM LIBRARY
END PROGRAM	LIST PROGRAM DESCRIPTION

Table II — Unique PIL functions

The functional characteristics of three selected PIL keys are given below.

INSERT CODE — This key allows code to be inserted at the location indicated by the light pen. All code after the indicated location is stepped down one line for each line of code to be inserted.

KEEP SOURCE PROGRAM — Pressing this key upon completion of the coding process stores the source program in the user's library. It is stored as a multi-page display and may be recalled using the SEEK AND SHOW key.

ADD PROGRAM TO LIBRARY — This key allows assembled programs to be added to the library from the standard output unit. Pressing the key will present a format display to the user with the request for the name of the program. The key number with which the program is to be associated and the status of the program key light are entered in the format. Upon entering the data required and pressing the ENTER key, the program will be placed in the user's library. The programmer is also required to describe his program. This information is associated with the LIST PROGRAM DESCRIPTION key.

Programming languages

Several programming languages are included in DOCUS at this time. These languages are FORTRAN, CODAP, DOL (Display Oriented Language), and Debugging Language. Using one of the programming key groups is very similar to pencil and paper coding. The principal advantages of on-line programming are not due directly to the on-line programming per se, but rather they result from associated system features:

- A program may be compiled or assembled immediately upon completion,
- On-line debugging aids are available,
- The program may be parameter tested immediately,

- Library facilities to keep the source and object programs are available,
- Facilities, via display, for dynamic on-line data entry at execution time are available,
- Results may be rapidly displayed in graphical formats,
- Program modification is simplified.

FORTRAN

The keys of the FORTRAN key group represent FORTRAN statements. When a function key is pressed the particular FORTRAN statement with appropriate blanks to be filled in is placed on the CRT screen. The operator then enters via the keyboard any variables necessary to complete the statement.

As each statement (after 16 lines) is completed, the page automatically rolls up 1 line so that coding may progress continuously. Each 32 lines is formatted as a page which may be recalled for reference or editing. The editing functions such as, INSERT CODE, DELETE CODE, and CHANGE CODE are available as part of the PIL key group which is in use simultaneously with a specific programming key group.

CODAP

One of the other programming key groups is for CODAP. The objective of this key group, designed primarily as a coding tutor, is to provide an on-line reference manual and a comprehensive source for utility routines. By having the instruction description and utility routine descriptions readily available more of the existing capability will be used. The keys associated with CODAP provide the following functions:

- Tab settings to specify coding columns for Labels, Op Code, Index, Address and Comments.
- Lists of mnemonics and machine code as a function of specific categories. Each category is displayed by pressing the specific key, e.g., SHIFT, FLOATING POINT ARITHMETIC, LOAD AND STORE, BRANCH, LOGICAL, INDEX, SELECT, SENSE, etc. When a category function key is pressed the current coding page is temporarily saved. It is returned by pressing the RESTORE PAGE key in the PIL key group.
- Descriptions for each instruction. Pressing the DESCRIBE INSTRUCTION key presents a multipage list display of all instructions. Selecting one entry from the list with the light pen and SELECT key shows a one page display which describes in detail the selected instruction.
- A function similar to the DESCRIBE INSTRUCTION is provided for utility routines.

The key is named DESCRIBE UTILITY ROUTINES.

- Assembly and coding instructions for the programming language which give the meaning of symbols used in the tutorial descriptions, directions about coding in the language and required control information for assembly.

Display oriented language

The third programming key group is for the Display Oriented Language (DOL). This language which is used to describe, create and manipulate displays and develop conversation mode procedures is represented by sample statements shown in Table III.

Controlling Devices on Console

LIGHTS ON	NAME LIGHT
LIGHTS OFF	DEFINE LIGHT GROUP
ALARM ON	TEST IF LIGHTS ON
ALARM OFF	READ CURSOR
SAVE LIGHTS	PROCESS LIGHT
	PEN MESSAGE
RESTORE LIGHTS	

Composition of Textual Displays

DEFINE AREA	MOVE AREA
LITERAL AREA	ROTATE AREA
DEFINE COMPOSITE	FILL AREA
DEFINE STRING	MOVE STRING
LITERAL STRING	

Composition of Graphical Displays

DRAW LINE	JOIN LINE
MOVE POINT	EXPAND FIGURE
ROTATE FIGURE	CONTRACT FIGURE
ERASE LINE	COMPOSE PICTURE

Table III — Sample of DOL statements

In the comprehensive form of DOL symbolic references can be made to information strings, areas, lines, pages, and displays. Full displays can be assembled from partial displays and static and dynamic displays can be mixed. Data entered into a format display can be extracted for use by a key program.

Debugging language

A Debugging Language is available for the programmer to assist him in key program preparation and checkout. Control and criteria data is entered via a format display. The output from the execution of the debugging operations is displayed on-line or written on the standard output unit for off-line printing. The language includes the functions shown in Table IV.

CLEAR
SNAPSHOT
STORAGE STATUS
TAPE DISPLAY
ENTER BREAKPOINT
PATCH
EXECUTE

Table IV — Debugging keys

The functional characteristics of two selected keys is given below.

SNAPSHOT — This function is used to produce a record of the contents of a storage area when the program being tested reaches a specific instruction. The same area may be displayed by each SNAPSHOT or a different area may be specified for each one. The programmer may have the option to specify the interval between SNAPSHOTS so that one need not be made for each pass of the program being tested. The point in the program at which the SNAPSHOT is made will be identified. The settings of the electronic switches, the processor console and registers and display console status lights are automatically included in SNAPSHOT.

EXECUTE — This key begins execution of a key program assigned to a key. The execution is done in a debugging mode with all breakpoints and patches effective. The display console is simulated since the program being debugged may use console features.

Application languages

The repertoire of application languages is essentially unlimited. To test some ideas about console usage, four languages of this type have been developed. These languages are simplified File Management Language, a Text Manipulation Language, a Computation Language and a Graphics Language. Only the former two will be discussed in this paper.

File management language

File Management consists of data base maintenance i.e., additions, deletions, and changes, and querying and output specification. In the development of the query capability consideration was given to ease of use by the operator. He does not have to have 1) an a priori knowledge of the data structure or content after it is initially defined or 2) training in computer programming and operation. Of course the more knowledgeable the user is about his data base the more efficient will be his use of the system. The system also eliminates redundant effort on the part of the user by allowing queries to be pre-stored.

File and record structure

Before either a query or file maintenance operation can be performed, it is necessary that the structure of

the file and record be defined. This is done only once for each file through the use of the DEFINE FILE STRUCTURE, DEFINE RECORD STRUCTURE and ACTIVATE keys. The two define keys present a display in which file and record structure parameters are entered. The ACTIVATE key stores this information in tables for later use with the query and file maintenance functions.

File maintenance

The File Maintenance function permits records to be added to the file, removed from the file, or changed. The particular file to be operated on is identified by operating the appropriate file key FILE 1, FILE 2, FILE 3, or FILE N. The FILE N key allows an identification to be selected from a list display. The particular functions are illustrated below:

FILE MAINTENANCE — Identifies the operation to be performed.

ADD RECORD — Adds to the specified file the record shown on the CRT screen. The record is checked against the record definition table.

DELETE RECORD — Removes from the specified file the record identified by sequence fields.

MODIFY RECORD — Permits a record shown on the CRT screen to be changed. The ADD FIELD and DELETE FIELD work in conjunction with this key by adding or deleting the field specified with the light pen and entered from the keyboard.

Query

One level logical queries may be formed and executed through the use of the keys associated with the query process. The file to be queried is identified by a file key in the same way as in the file maintenance function. The key QUERY identifies the operation to be performed. Sequence fields of the record are identified by the keys SELECT BY FIELD 1, 2, ..., 5. The SELECT BY FIELD N presents a list display showing the other fields in a record. Fields may be selected from this list by the light pen. Following a field selection, criteria values are entered in a format display. After the criteria is entered a logical selection may be made by operating the AND, OR or BUT NOT keys. After a logical selection, a field selection must be made and criteria values entered.

The specification of a query is terminated by the operation of an output key. These keys are: 1) CRT which shows the retrieved records on the CRT, 2) PRINTER which prints the retrieved records and 3) SUBSET DATA BASE which makes a new file of the retrieved records in their original format. This file may be saved or subsequently queried.

Text manipulation language

The Text Manipulation Language is designed to edit,

display, tag, and organize textual data. At the console the user may operate keys to create statements which operate on a visual display.

Control functions

are: ROLL/STOP/COMMENCE
 Verbs are: SPACE/REPLACE/INSERT/
 DELETE/COPY/TAG/TRANPOSE
 Nouns are: MARK/CHARACTER/WORD/
 CLAUSE/SENTENCE/LINE/
 PARAGRAPH/PAGE/NOTE

ROLL is a typical control function which moves lines of text up and down the CRT screen. It is desirable to have some control over the rate at which a line moves. For this purpose the CRT cursor can be used. The rate of motion is controlled by the Y coordinate of the cursor on the CRT screen as illustrated in Figure 6.

With the cursor in the center of the vertical axis there is no motion. With the cursor at the top of the screen, motion is maximum in the forward direction. When the

cursor is placed in the lower half of the screen motion is reversed. In general, the rate of motion is proportional to the distances C/Y with + being forward and - being backward. Pressing STOP, or COMMENCE overrides the speed control.

Editing may be done when rolling is stopped by pressing function keys, light pen selection, and alphanumeric type-in as appropriate. Combinations of a Verb and a Noun are used to create operational statements. For example, in Figure 7 a word may be inserted as follows:

- 1) Press INSERT
- 2) Press WORD
- 3) Light Pen MANIPULATION
- 4) Type in OVERLAY
- 5) Press ENTER

The system expands the text to accommodate the new word at the specified location.

Portions of the existing text may be used to build new

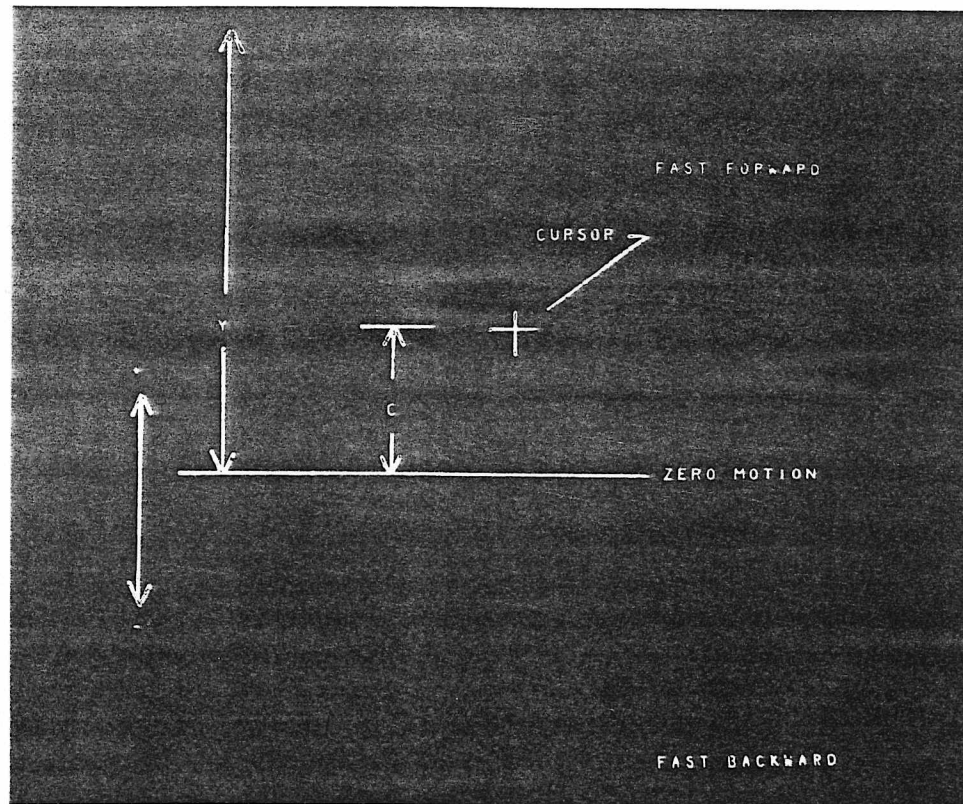


Figure 6 — Roll rate control

text. In Figure 7 the word TEXT may be copied in a new location by:

- 1) Press COPY
- 2) Press WORD
- 3) Light pen the word TEXT
- 4) Light pen the word before the location where TEXT is to go.

After all editing has been done, the user may press OUTPUT EDITED TEXT to obtain, for review, an updated version of his input text. A format display allows him to select the medium on which he wishes his outputs. Similarly, the user may choose to have the text written out by pressing OUTPUT TEXT.

ACKNOWLEDGMENT

The authors wish to acknowledge the support provided by Mr. William Moore and Mr. Sam DiCarlo of the Rome Air Development Center. Their keen insight into the problems of man/machine communications and

their early recognition of the need for exploratory development of generalized CRT console software has made the DOCUS project possible. We also wish to thank the DOCUS project staff for the programming and many hours of night and weekend debugging, and especially Mr. George Collins for his creative contributions to the project and constructive review of this paper.

REFERENCES

- 1 W F BAUER W L FRANK
 DODDAC — An integrated system for data processing, interrogation and display
 Proceedings EJCC December 1961
- 2 CULLER-FRIED
 An on-line computing center, RADC-TDR-63-160
 July 1963 AD 414564 Contract AF30(602)-2762
 Thompson Ramo Wooldridge Inc
- 3 The TRW two station, on-line scientific computer, RADC-TDR-64-392
 December 1964 AD 609720 Contract AF30(602)-3097
 TRW/Space Technology Laboratories

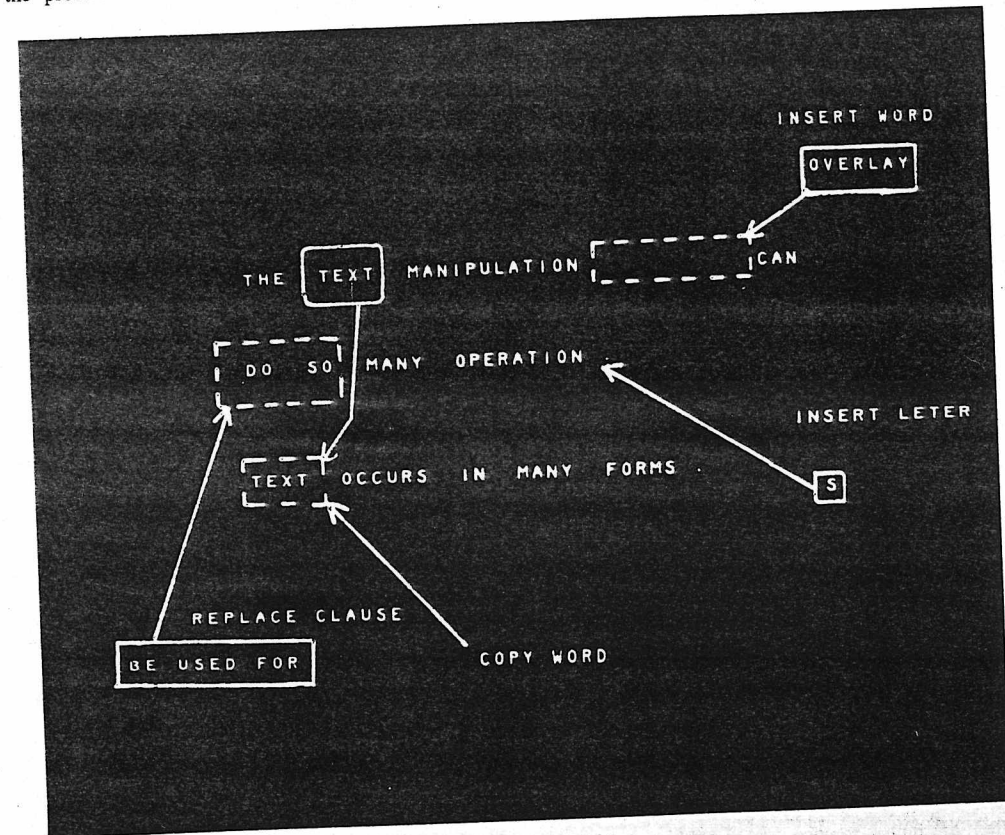


Figure 7 — Text manipulation

REFERENCES (Continued)

- 4 *National military command system support center display system*
Volumes I II III Contracts DA-49-146-XZ-211 and DCA-16
- 5 S L KAMENY G P DEFLORIO A H VARHAUS
General purpose display system
September 1964 SP-1699 AD 451859
System Development Corporation
- 6 E BENNETT E HAINES J SUMMERS
AESOP: A prototype for on-line user control of organizational data storage, retrieval and processing
Proceedings FJCC May 1965
- 7 I E SUTHERLAND
Sketchpad: A man-machine graphical communication system
January 1963 Report No 296 and SJCC 1963
Lincoln Laboratory
- 8 T R ALLEN J E FOOTE
Input/output software capability for a man-machine communications and image processing system
FJCC 1964 General Motors Corporation
- 9 ALPINE
Graphical design system
IBM motion picture
- 10 *Advanced programming developments: a survey: ESD-TR-65-171*
February 1965 Computer Associates Incorporated
- 11 T B STEEL
Implicitly programmed systems
System Development Corporation Tech Memo
TM-1663/000/00 December 1963
- 12 F A DION
A user oriented information processing system
RADC-TDR-64-1963 June 1964 Rome Air
Development Center
- 13 E A AVAKIAN W F JENSION
Voice response and visual display techniques for on-line information handling systems
Information Display November/December 1964
- 14 F B THOMPSON
The semantic interface in man-machine communication
September 1963 RM63TMP-35 Contract Nonr-4101(00)
General Electric Company
- 15 D B PARKER
Graphical communications in an on-line system
Control Data Corporation
- 16 T H TACK
Software for display consoles
February 1964 Proceedings SID 3rd Symposium
- 17 H S CORBIN G J STOCK
On-line querying via a display console
Proceedings SID 4th Symposium

A language and model for computer-aided design

by N. J. DENIL
IBM Systems Development Division
Kingston, New York

INTRODUCTION

This paper describes Design Language I (DLI) — a prototype computer-aided design system. DLI addresses the creative design of 3-dimensional structure, using a display console for communication with the user.

The ultimate objective of a design system is to provide a mechanical designer with a display console on which he may design parts and assemblies, do engineering analyses on these designs, modify them, and finally produce drawings and other engineering documentation. The design process in this ultimate system is characterized by changes — solution to a design problem is worked into a final design by a succession of changes and refinements.

Earlier systems, notably Sketchpad¹ and Sketchpad III² have treated the same problem area in two and three dimensions. Both used a display console as a sketching medium and the computer to form a model of the information in the sketch. DLI treats both the console input and the modelling problem in a different manner. Pushbuttons, though available, are replaced as a principal input by a written language. The DLI model approaches the structure representation from a different point of view.

The system is intended to provide the user with a language with which to describe and manipulate parts consisting of geometrical structures. Underlying the language is a processor which interprets what the user says to the extent of its stored information of geometry and builds an internal representation of what the user is talking about. The system provides the user the capability to perform the following functions:

1. Define elements to any depth. The mere existence of an item and its name are a sufficient quantum of information — additional attributes and definers can be added at any time.
2. Find out how an element is currently defined or interconnected.
3. Change any element by adding new definers which override some or all of the previous ones; being informed by the system of the changes which are taking place.

4. On the display, view his structure from any vantage and be able to sketch, identify things, and input new information in any convenient view.

The main subject of this paper is a description of the processor and model used to provide these capabilities. Before describing that, however, a view of the external appearance of the system is in order. The next section gives such an overview. Then the internal operations are described, followed by remarks on the system and its extension.

EXTERNAL OPERATING CHARACTERISTICS

DLI was implemented on an IBM 7044 Data Processing System equipped with a 7226 graphic unit consisting of a display console and recorder. The console was equipped with a light pen, function keys, and an alphameric keyboard.

The user communicates with the system via the graphic console — on it are displayed (1) the language with which he instructs the system, (2) views of the object he is building, and (3) messages from the system to the user.

The prototype DLI system handles design of 3-dimensional objects consisting of points, straight lines, and planes. Additional elements are scalars (holders for single numbers), sets (collections of elements), views (2-dimensional displays of elements) and modes (holders for definers to be applied to each element created when the mode is active).

The console

The console screen is arranged as shown in Figure 1. Across the top is displayed the language command which the user constructs to operate the system. Along the left side of the screen are displayed the vocabulary choices which are appropriate at any particular time, and adjacent to this column is the control vocabulary — the command punctuation and termination characters — which cause actions to be performed. Below the command are two additional areas — the alphameric buffer, where characters entered from the keyboard are displayed, and the process/idle flag, which indicates whether the 7044 is processing or waiting. In the lower